Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vjeran Rađa

Pseudospectral methods for boundary value problems

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Bojan Orel

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Vjeran Rađa, z vpisno številko **63100443**, sem avtor diplomskega dela z naslovom:

Pseudospectral methods for boundary value problems

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Bojana Orla,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. julija 2015 Podpis avtorja:

I would like to express gratitude to my family, friends, girlfriend and mentor for patience and support while I was writing the thesis.

Contents

Povzetek

Abstract

1 Preface **1**

2 Theory **3**

2.1 Differential equations 3

2.2 Interpolation 6

2.3 Cardinal functions 8

3 Spectral methods **11**

3.1 Idea 12

3.2 Convergence of spectral methods 13

3.3 Choice of basis functions 14

3.4 Boundary Conditions 16

3.5 Basis recombination 17

4 Practical application **21**

4.1 Introduction to software 21

4.2 Computer program 30

5 Conclusion **35**

Table of contents

Povzetek

Namen diplomske naloge je preučiti psevdospektralne metode in implementirati algoritem za reševanje dvotočkovnih robnih problemov. Drugi cilj diplomske naloge je implementirati ta algoritem v okviru aplikacije, ki deluje za celo množico robnih problemov v dveh točkah z uporabo programskega jezika Python.

Najprej je postavljena teoretična baza: uvod v diferencialne enačbe, robni problemi ter še nekoliko konceptov potrebnih za razumevanje psevdospektralnih metod. Potem je predstavljena teoretična ideja psevdospektralnih metod in teoretičen način implementacije algoritma ter na koncu praktična realizacija in predstavitev spletne aplikacije za reševanje robnih problemov v dveh točkah.

Diplomska naloga je razdeljena v tri logična dela: Theory, Spectral methods in Practical application.

V prvem poglavju diplomske naloge so razloženi osnovni pojmi vezani za temo naloge. Začetek poglavja je rezerviran za diferencialne enačbe: navadne diferencialne enačbe, parcialne diferencialne enacbe, začetne probleme in robne probleme. Prve tri kategorije so predstavljene z nekoliko stavkov ter enim osnovnim primerom iz vsake kategorije. V podpoglavju robnih problemov so na kratko opisani Dirichletovi, Neumannovi in Robinovi robni pogoji kateri so pomembni za praktičen del diplomske naloge. Za diferencialnim enačbam sledi interpolacija, ki je osnovno "orodje" v izračunu dvotočkovnih robnih problemov z psevdospektralno metodo. Na koncu poglavja so na kratko opisane kardinalne funkcije s sliko kardinalne funkcije iz praktičnega

dela diplome.

Drugi del diplomske naloge se nanaša na spektralne metode. Na začetku poglavja je na kratko razložena ideja psevdospektralnih metod. Osnovna ideja je, da približek iskane funkcije lahko izračunamo kot linearno kombinacijo $N+1$ baznih funkcij z enačbo

$$u(x) \approx u_N(x) = \sum_{n=0}^{N} a_n \phi_n(x).$$

Sledi podpoglavje konvergence spektralnih metod katero vsebuje definiciji o algebrainoj konvergenci spektralnih metod. Round-off plateau je podpoglavje namenjeno napaki zaokroževanja računalnika katera približno znaša 10^{-16}. Napaki sledi način izbire baznih funkcij, ki je baziran na Lagrangeovi interpolacijski formuli. Lagrangeova interpolacijska formula kaže, da v polinom stopnje N lahko vstavimo $N+1$ točk

$$P_N(x) \equiv \sum_{j=0}^{N} f(x_j)\phi_j(x).$$

in je posplošitev zgornje enačbe. ϕ_j je kardinalna funkcija. Na ta način zagotovimo, da kardinalna funkcija izgine v vsem interpolacijskim točkami razen v točki $x = x_j$. Na koncu poglavja imamo še razloženo idejo rekombinacije robnih pogojev iz nehomogenih v homogene kar olajšuje izračun funkcije z psevdospektralno metodo na način, da ni potrebno posebej računati robne pogoje.

V prvem delu četrtega poglavja so predstavljene knjižnice v programskem jeziku Python, skupaj z Djangom, ki je framework za izdelavo spletnih aplikacij in spletnih strani. Python se uporablja zaradi svoje podobnosti Matlabu, zaradi dobrih matematičnih knjižnic Numpy, ScyPy in SymPy ter zaradi lahke implementacije v obliki spletne aplikacije kar zagotavlja prenosljivost, neodvisnost od operacijskega sistema ter omogoča enostavno razširitev.

Numpy in ScyPy sta knjižnici za računske operacije z matrikami. Sympy se uporablja za simbolično matematiko oziroma bolj natančno za interpretacijo matematičnih izrazov vpisanih formo na prvi strani spletne aplikacije, da

jih program v ozadju lahko izračuna. Temu sledi razlaga samega algoritma izračuna v podpoglavju "A MATLAB diferentiation matrix suite". V tem poglavju so razložene podrobnosti algoritma, formule katere se uporabljajo za izračun in predstavljen je način na kateri program obravnava Dirichletove, Neumannove in Robinove robne pogoje. Po tem, v poglavju Examples se nahajata dva primera delovanja programa z slikami izgleda aplikacije pri vnosu in izračunu vnešenega robnega problema. Na koncu se še nahaja napaka pri izračunu glede na stvarno rešitev predstavljenega problema. Program je zmožen izračuna rešitve in napake, če uporabnik obkljuka polje za vnos rešitve. Velja omeniti, da šele pri 150 točkah program kaže bolj natančne rezultate.

Ključne besede: psevdospektralne metode, robni problemi, Python

Abstract

The purpose of this thesis is to study pseudospectral methods and to implement the algorithm that solves two point boundary value problems by pseudospectral methods. Another purpose of this thesis is to implement that algorithm into an application for solving two point boundary value problems.

The first part of the thesis presents theoretical concepts needed to understand the pseudospectral methods such as: differential equations, boundary value problems and few more. This is followed by the theoretical idea of pseudospectral methods and algorithm in theory. The last part consists of practical implementation and explanation of how application works followed by few examples.

Keywords: pseudospectral methods, boundary value problem, Python

Chapter 1

Preface

As complexity of numerical calculations in today's world increases we need reliable, fast and memory efficient ways to calculate them. Pseudospectral methods are applicable for solving partial and ordinary differential equations in the majority of cases.

Today these equations are mostly used to help solving problems in fluid dynamics, weather prediction and optimal control problems.

In Chapter 2 the theoretical concepts that are essential for understanding pseudospectral methods are explained. That includes differential equations, both ordinary and partial, and initial value problems. Topic of boundary value problem is explained in more detail than previous topics as they are important for this thesis. Chapter two concludes with interpolation and cardinal functions.

Chapter 3 is dedicated to spectral methods. The first part of the chapter explains the idea of spectral methods and precision of calculation i.e the computer round-off error. Further on the choice of basis functions is explained as well as usage of basis functions in practice. The end of the chapter is dedicated to the explanation of boundary condition types from numerical mathematics point of view and basis recombination method which is used to help in calculation of boundary conditions in the practical part of the thesis.

Chapter 4 introduces the software used in the practical part of the thesis

as well as the algorithm used to solve two point boundary value problems via web application. The chapter ends with the introduction of the web application which solves two point boundary value problems and shows some examples of how the program works.

Chapter 2

Theory

2.1 Differential equations

Differential equations play an extremely important and useful role in applied mathematics, engineering, and physics, and much mathematical and numerical machinery has been developed for the solution of differential equations. In a section below some definitions and types of differential equations are presented.

A differential equation is an equation that involves the derivatives of a function as well as the function itself.

When applied, the functions usually represent physical quantities, the derivatives represent their rates of change, and the equation defines a relationship between the two. According to the solution functions differential equation (DE) can be classified as an ordinary differential equation (ODE; the solution function has a single variable) and a partial differential equation (PDE; multivariable solution function).

2.1.1 Ordinary differential equations

Ordinary differential equations are those in which there appear derivatives with respect to only one independent variable. A simple example is:

$$\frac{df(x)}{dx} = \alpha f(x), \tag{2.1}$$

where α is a constant [1]. A second example, that is a bit more applicable, is one where a trajectory of projectile launched from a cannon follows a curve determined by an ODE that is derived from Newton's second law of motion:

$$m\frac{d^2x(t)}{dt^2} = f(x(t)). \tag{2.2}$$

2.1.2 Partial Differential Equation

A partial differential equation is a differential equation that contains unknown multivariable functions and their partial derivatives.
An example of PDE would be the heat equation:

$$\frac{\partial u}{\partial t} = h^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \tag{2.3}$$

Partial differential equations are used to formulate problems involving functions of several variables, and they are used to create a relevant computer model.

PDEs can be used to describe a wide variety of phenomena such as sound, heat, electrostatics, electrodynamics, fluid flow, elasticity, or quantum mechanics. These seemingly distinct physical phenomena can be formalised similarly in terms of PDEs. Just as ordinary differential equations often model one-dimensional dynamical systems, partial differential equations often model multidimensional systems [3]. PDEs can also be classified according to the type of differential operator involved.

2.1.3 Initial Value Problem

Initial value problem (IVP) is an ordinary differential equation together with specified value, called the initial condition, for example

$$y' = f(t, y), \quad y(t_0) = y_0, \tag{2.4}$$

where f is a known function of t and y, and t_0 and y_0 are given values. The object of solving this problem is to find y as a function of t [8].

A popular example of initial value problem is *bacterial population growth*:

$$\frac{dy}{dt} = kt, \quad y_0 = a, \tag{2.5}$$

where y is a number of bacteria at time t and y_0 is the initial size of the bacteria population at $t = 0$.

2.1.4 Boundary Value Problem

In contrast to initial value problem, boundary value problem (BVP) has its conditions specified more than one point (also called *boundaries*) of the independent variables in the equation.

Boundary value problems mostly arise from physical problems. In the mathematical field of differential equations, boundary value problems are differential equations with a set of restraints at two (or more) values of the independent variable called the boundary conditions. There are a few types of boundary conditions [6]. In the practical part the following are used:

1. The solution has some particular value at the end point or along a boundary (Dirichlet condition);

2. The derivative of the solution equals a particular value at the end point or in the normal direction along a boundary (Neumann condition);

3. Weighted combination of Dirichlet and Neumann boundary condition (Robin condition);

In general, boundary value problems will reduce, when discretized, to a set of linear (and sometimes nonlinear) equations. To apply them properly, boundary value problems should be well posed. In other words, they have to (according to Jacques Hadamard [5]) satisfy three properties:

1. A solution has to exist;
2. The solution has to be unique;
3. The solution's behavior changes continuosly with the additional (initial or boundary value problem's) conditions.

The relationship between physics and mathematics is important in boundary value problems, because it is not always possible for a solution of a differential equation to satisfy arbitrarily chosen conditions; but if the problem represents an actual physical situation, it is usually possible to prove that a solution exists, even if it cannot be explicitly found [7].

2.2 Interpolation

Interpolation is a method of constructing the new function called interpolant which belongs to a certain closed set of functions (most often polynomials). Interpolant agrees with values of a given function at a set of interpolation points.

In general, this technique involves the construction of a function $L(x)$ called the interpolant which agrees with f at the points $x = x_i$ and which is then used to compute the approximate values of a given function at intermediate points. There are various methods of calculating the interpolant of a given function.

By taking two neighbouring points the one between them can be calculated and is referred to as *linear interpolation*. It is the simplest and quickest interpolation method although not very precise. *Polynomial interpolation* gives better results. Given some points we have to find the polynomial that

goes exactly through these points. Although polynomial interpolation gives better results than linear interpolation it is computationally expensive compared to linear. Furthermore, polynomial interpolation may exhibit oscillatory effects, especially at the end points (Runge's phenomenon).

It seems plausible that if we distribute the interpolation points evenly over an interval $[a, b]$, then the error in $P_N(x)$ should $\to 0$ as $N \to \infty$ for any smooth function $f(x)$. But, in his paper [13] Runge showed that, when using polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points, higher polynomial degrees do not always improve accuracy. Runge proved that the middle of the interval is not the problem and that big errors occur near endpoints. This suggests that we should space the grid points relatively far apart near the middle of the interval where we are getting high accuracy anyway, and increase the density of points as we approach the endpoints. To answer the question of best point distribution we have to look at [11]:

Theorem 1 ***(CAUCHY INTERPOLATION ERROR THEOREM)*** *Let $f(x)$ have at least $(N+1)$ continuous derivatives on the interval of interest and let $P_N(x)$ be its Lagrangian interpolant of degree N. Then*

$$f(x) - P_N(x) = \frac{1}{[N+1]!} f^{(N+1)}(\xi) \prod_{i=0}^{N} (x - x_i) \tag{2.6}$$

for some ξ on the interval spanned by x and the interpolation points. The point ξ depends on the function being approximated, upon N, upon x, and upon the location of the interpolation points $x_i, \quad i = 0, ..., N$.

Proof is in [14].

If we want to optimize Lagrangian interpolation, there is nothing we can do about the $f^{(N+1)}(\xi)$ factor since it depends on the specific function being approximated, but the magnitude of the polynomial factor depends upon our choice of grid points. It is evident that the leading coefficient of

$$\prod_{i=1}^{N} (x - x_i), \tag{2.7}$$

is 1, independent of the grid points, so the question becomes: What choice of grid points gives a polynomial (with leading coefficient 1) which is as small as possible over the interval spanned by the grid points? By a linear change of variable, the interval $[a, b]$ can always be re-scaled and shifted to $[-1, 1]$.

2.3 Cardinal functions

Cardinal functions $\Phi_j(x)$ for a given type of interpolation and for the set of interpolation points x_i are defined by the requirement that

$$C_j(x_i) = \delta_{ij} \qquad i, j = 1, ..., N \tag{2.8}$$

where δ_{ij} is the Kronecker delta symbol defined by

$$\delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j. \end{cases} \tag{2.9}$$

A cardinal function has the property of being non-zero at exactly one of the interpolation points. That implies that for arbitrary $f(x)$ the function defined by

$$\tilde{f}(x) \equiv \sum_{j=-\infty}^{\infty} f(x) C_j(x) \tag{2.10}$$

interpolates $f(x)$ at every point of the grid.

f	x
0.0	-1.0
-0.704160951075	-0.66666667
-0.841335162323	-0.33333333
0.0	0.0
0.168267032465	0.33333333
-0.100594421582	0.66666667
0.0	1.0

Table 2.1: Data used to plot cardinal function in figure 2.1.

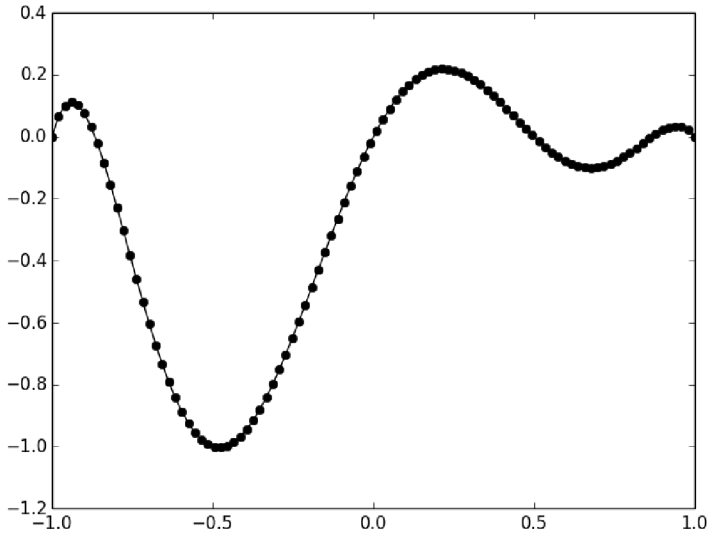

Figure 2.1: Example of a cardinal function taken from a practical part of this thesis.

Chapter 3

Spectral methods

Spectral methods is a collective name for spatial discretization methods for solving differential equations that rely on an expansion of the solution as a linear combination of basis functions. These basis functions usually have global support on the domain, and spatial derivatives are defined in terms of deriatives of these basis functions. The coefficients pertaining to the basis functions can be considered as a spectrum of the solution, which explains the name for the method. Due to global nature of the basis functions, spectral methods are usually global methods, i.e. the value of a derivative at a certain point in space depends on the solution at all the other points in space, and not just the neighbouring grid points. Due to this fact, spectral methods usually have a very high order of approximation. They usually give the exact derivative of a function, the only error is due to the truncation of a finite set of basis functions [10].

Spectral methods can be divided into two broad categories: *collocation* and *non-collocation*.[1] At first glance, there is no obvious relation between the collocation method and the alternatives that use weighted integrals. Although *pseudospectral* method is in a *collocation* method category, it is equivalent to the weighted integrals method if we evaluate the integrals of the

[1] While this classification is not entirely correct because some algorithms mix ideas from both the *collocation* and *non-collocation* category, it provides broader insight on the topic.

latter by numerical quadrature with $(N+1)$ points, or more precisely, by using Gaussian quadrature points for the integrals of the Galerkin method. Historically, the *non-collocation* methods were developed first and the label *spectral* is sometimes used in a narrow sense as a collective tag for the *non-collocation* methods. Furthermore, *spectral* has other meanings in fields such as time series analysis and functional analysis. This is the reason why the collocaton-based methods are now commonly called *pseudospectral* [11].

3.1 Idea

The basic idea is to assume that the unknown $u(x)$ can be approximated by a linear combination of $N+1$ *basis functions* $\phi_n(x)$:

$$u(x) \approx u_N(x) = \sum_{n=0}^{N} a_n \phi_n(x). \tag{3.1}$$

The equation

$$u'' + u'q + ur = f(x) \tag{3.2}$$

can be restated as

$$Lu = f(x), \tag{3.3}$$

where L is the operator of the differential equation. When the series (3.1) is substituted into (3.3) the result is the so-called *residual function* defined by

$$R(x; a_0, a_1, ..., a_N) = Lu_N - f. \tag{3.4}$$

Since the residual function $R(x; a_0, a_1, ..., a_N)$ is identically equal to zero for the exact solution, the challenge is to choose the series coefficients a_n so that the residual function is minimized. The different spectral and pseudospectral methods differ mainly in their minimization strategies.

The coefficients a_n of a pseudospectral approximation to the solution of a differential equation are found by requiring that the residual function satisfies:

$$R(x_i; a_0, ..., a_N) = 0, \qquad i = 0, 1, ..., N \tag{3.5}$$

meaning that the differential equation is exactly satisfied at a set of points known as *collocation* or *interpolation* points. Presumably, as $R(x; a_0, ..., a_N)$ is forced to vanish at an increasingly large number of discrete points, it will be smaller and smaller in the gaps between the collocation points so that $R \approx 0$ everywhere in the domain, and therefore $u_N(x)$ will converge to $u(x)$ as N increases. The accuracy of pseudospectral methods is only a little bit poorer than that of the *non-collocation* methods. But if we consider the fact that these methods offer greater simplicity and computational efficiency, the benefits of the former emerge.

3.2 Convergence of spectral methods

The spectral algorithm should be chosen to maximize the algebraic convergence order for a given problem. The definition of algebraic index of convergence is [11]:

Definition 1 *The* algebraic index of convergence *is the largest number k for which*

$$\lim_{n\to\infty} |a_n| n^k < \infty, \tag{3.6}$$

where a_n are the coefficients of the series. (For a Fourier series, the limit must be finite for both the cosine coefficients a_n and the sine coefficients b_n.)

Definition 2 *If the algebraic index of convergence k is unbounded, or in other words, if the coefficients a_n decrease faster than $1/n^k$ for any finite power of k, then the series is said to have the property of* ***infinite order****,* ***exponential****, or* ***spectral*** *convergence.*

3.2.1 Round-off Plateau

The *Round-off Plateau* or the *Machine* ϵ is a point where the algebraic order of convergence may be defeated simply because the machine rounds off numbers at some point. Or more formally [11]: Let a_{max} denote the maximum

absolute value of the spectral coefficients a_j for all j. Let ϵ denote a constant proportional to the *round-off error* or *machine epsilon*, typically around 10^{-16} on most computers[2]. Then when the exact coefficients fall below ϵa_{max}, spectral algorithms including interpolation will compute round-off-corrupted coefficients that will flatten out at roughly $a_n \approx \epsilon a_{max}$ for all sufficiently large n. Roughly means that coefficients in the *Roundoff Plateau* fluctuate randomly about the indicated magnitude as shown in 3.1. It is noteworthy that individual coefficients a_k can be computed correctly. The problem occurs when the coefficients are added to the series.

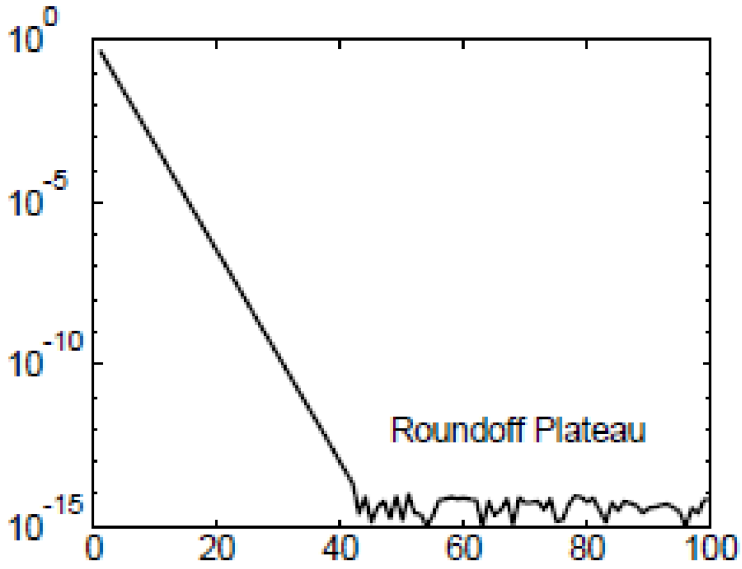

Figure 3.1: Graphical display of Round-off plateau.

3.3 Choice of basis functions

The important property of the *basis functions* are ease of computation and rapid convergence. In other words, it would be preferred if any solution

[2]At my computer tests showed that that point is $2.2204460492503131 * 10^{-16}$. I use Python programming language for the practical part of this thesis.

could be represented with arbitrarily high accuracy by taking the truncation treshold to be sufficiently large.

In general, one may fit any $N+1$ points by a polynomial of N-th degree via Lagrange Interpolation Formula:

$$P_N(x) \equiv \sum_{j=0}^{N} f(x_j)\phi_j(x). \tag{3.7}$$

where ϕ_j is cardinal function defined by (2.8).

The N factors insure that $\phi_j(x)$ vanishes at all the interpolation points except x_i and $\phi_i(x)$ is equal to 1 at the interpolation point $x = x_i$. The cardinal function representation is not efficient for computation, but it does give a proof-by-construction of the theorem that it is possible to fit an interpolating polynomial of any degree to any function. Although the interpolating points are often evenly spaced, no such restriction is inherent in (3.7). The formula is valid even if the $\{x_i\}$ are unevenly spaced or out of numerical order.

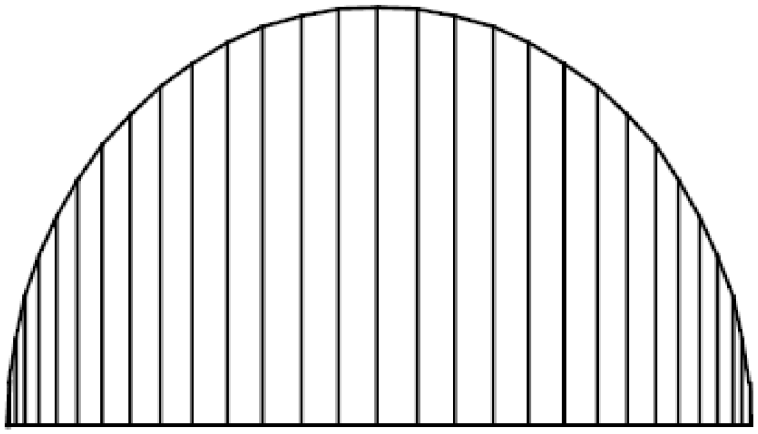

Figure 3.2: Graphical construction of the unevenly spaced Chebyshev interpolation grid.

In theory, Chebyshev, Fourier and cardinal functions can be used as basis functions. The practical part of this thesis makes use of the cardinal function basis calculated from the first derivative of Chebyshev polynomials

where N is the degree of Chebyshev polynomial used to interpolate function being calculated with a program. This way basis functions are homogeneous and individually satisfy boundary conditions so there is no need to calculate boundary conditions separately from inner collocation points.

Chebyshev nodes are calculated from the equation

$$x_j = \cos\left(\frac{k\pi}{N}\right), \quad j = 0, ..., N. \tag{3.8}$$

Cardinal function is then calculated using following formula:

$$\phi_j(x) = \frac{(-1)^j}{c_j} \frac{1-x^2}{(N-1)^2} \frac{T_N'(x)}{x - x_j}. \tag{3.9}$$

Here $c_0 = C_{N-1} = 2$, $c_i = 1$ for $i = 1, \ldots, N-2$, and $T_N(x)$ is the Chebyshev polynomial of the first kind, described in [12] . This is explained in more detail in 4.1.6 .

3.4 Boundary Conditions

Boundary conditions are divided into two broad categories [11]: behavioral and numerical. Periodicity is a behavioral condition: it demands that the solution $u(x)$ have the property of $u(x) = u(x+2\pi)$ for any x, but this behavior does not impose any specific numerical value upon $u(x)$ or its derivatives at any particular point. Another behavioral condition is that of being bound and infinitely differentiable at a point where the differential equation is singular. In contrast, conditions such as $u(-1) = 3$ and $u(1) + du/dx(0) = 1.7$ are numerical.

The significance of these categories is that it is almost always necessary to explicitly impose numerical boundary conditions. In contrast, behavioral conditions may be satisfied implicitly by choosing basis functions such as that each has the required property or behavior.

When the boundary conditions are such that $u(x)$ is periodic in x, then it is best to choose sines and cosines of a Fourier series since each basis function is periodic, which implies that their sum will be periodic too. Consequently,

it is unnecessary to explicitly impose periodicity on the numerical solution. Our choice of basis function has implicitly satisfied the boundary conditions.

Numerical boundary conditions must be explicitly imposed. There are two strategies for satisfying boundary conditions:

- *Boundary-bordering* - reducing the number of collocation conditions on the residual of the differential equation, and using the rows of the pseudospectral matrix to explicitly impose the constraint.

- *Basis recombination* - modifying the problem so that the boundary conditions of the modified problem are homogeneous and then altering the basis set, so that the basis functions individually satisfy these conditions.

3.5 Basis recombination

If the problem

$$u'' + u'q + ur = f \tag{3.10}$$

has inhomogeneous boundary conditions it may be replaced by an equivalent problem with homogeneous boundary conditions, so long as they are linear. First we have to choose a simple function $B(x)$ which satisfies the inhomogeneous boundary conditions. We can define a new variable $v(x)$ and a new forcing function $g(x)$ via

$$\begin{aligned} u = v + B(x) = v + (Kx + n) \\ v'' + v'q + vr + qK + (Kx + n)r = f \\ g = f - qK - (Kx + n)r. \end{aligned}$$

The shift function B is arbitrary except for the constraint that it must satisfy all the inhomogeneous boundary conditions.

Example

Algorithm for basis recombination on a two point boundary ordinary differential equation with the mixed conditions

$$u_x(-1) = \gamma,$$
$$\frac{du}{dt}(1) - u(1) = \delta$$

Let's assume

$$B(x) = (Kx + n) \tag{3.11}$$

and choose two boundary coefficients by demanding that $B(x)$ satisfy the boundary conditions. Let's modify the problem so boundary conditions of that problem become homogeneous:

$$\frac{dB}{dx}(-1) = \gamma \quad \leftrightarrow \quad K = \gamma \tag{3.12}$$
$$\frac{dB}{dx}(1) - B(1) = \delta \quad \leftrightarrow \quad K - (K + n) = \delta \tag{3.13}$$
$$\rightarrow n = -\delta \tag{3.14}$$
$$\rightarrow B(x) = \gamma x - \delta \tag{3.15}$$

When $B(x)$ is found and the problem is transformed to finding $v(x)$ we can start with *basis recombination*. By choosing linear combinations of the original basis functions the new combinations of basis functions are calculated and each of those combinations individually satisfy the homogeneous boundary conditions.
For example, suppose that

$$v(-1) = v(1) = 0. \tag{3.16}$$

Equation (3.9) ensures that $v(x)$ is equal to zero at those two points.

Collocation points are computed from Chebyshev grid formula

$$x_i = \cos\left(\frac{\pi i}{N-1}\right) \qquad i = 1, 2, ..., (N-2). \tag{3.17}$$

Furthermore

$$v(x) \approx \sum_{n=2}^{N-1} b_n \phi_n(x). \tag{3.18}$$

If we define a column vector $\mathbf{b}$ of dimension $(N-2)$ whose i-th element is $\mathbf{b}_{i-1}$, then the differential equation becomes the matrix problem

$$H\mathbf{b} = G, \tag{3.19}$$

where

$$H_{ij} \equiv \phi''_{j+1}(x_i) + q(x_i)\phi'_{j+1}(x_i) + r(x_i)\phi_{j+1} \qquad i,j = 1,...,N-2, \tag{3.20}$$

where the double subscript x denotes double x-differentiation and

$$G_i \equiv g(x_i) \qquad\qquad i = 1,2,...(N-2) \tag{3.21}$$

After solving (3.19) to compute the coefficients of the ϕ-series for $v(x)$, the sum is converted from ordinary to cardinal

$$v(x) \equiv \sum_{j=2}^{N-1} b_j \phi_j(x), \tag{3.22}$$

ϕ_j is defined in (3.9).

In the general case, the great advantage of *basis recombination* is conceptual simplicity. After shifting to the modified basis set, we can ignore the boundary conditions and concentrate on solving the differential equation. All the rows of differentiation matrix are derived from setting the residual of the differential equation equal to zero at a given collocation point.

Chapter 4

Practical application on a computer program

In this chapter, the application of theory on a program which solves two-point boundary value problem in the form of:

$$u''(x) + q(x)u'(x) + r(x)u(x) = f(x), \qquad -1 < x < 1 \tag{4.1}$$

$$a_+u(1) + b_+u'(1) = c_+, \tag{4.2}$$

$$a_-u(-1) + b_-u'(-1) = c_-, \tag{4.3}$$

is discussed.

4.1 Introduction to software and the libraries used

In order to make the application straightforward, easy to use on general examples and modal, Python programming language was used [17]. Python offers SciPy [19] package which incorporates several libraries which use the code similar to Matlab's for solving mathematical problems. Because some crucial parts of the program use functions presented in "A MATLAB differentiation matrix suite" [21] and interpretation of user input should work

for a general example, it was decided to use the library for symbolic mathematics. Python offers Sympy [24] module with syntax similar to Wolfram Mathematica. That made the choice of programming language an easy decision. It was furthermore decided that it was best to make the user interface in the form of a web page. CSS and HTML languages are straightforward, stable and work in every browser equally, which offers cross-platform usage of my application without having to adjust the code to different operational systems. The choice of Django framework makes the application easier to change, and it makes it easier to expand its functionalities in potential future releases.

4.1.1 Python

Python[17] is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both small and large scale [18]. In my work I have used Python version 2.7.

4.1.2 NumPy

NumPy [22] is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays. It is open source and has many contributors [23].

4.1.3 SciPy

SciPy [19] (pronounced “Sigh Pie”) is an open source Python library used by scientists, analysts, and engineers doing scientific computing and technical computing. SciPy contains modules for optimization, linear algebra,

integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas, and SymPy. The SciPy library is currently distributed under the BSD license, and its development is sponsored and supported by an open community of developers [20].

4.1.4 SymPy

SymPy [24] is a Python library for symbolic computation. It provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web as SymPy Live or SymPy Gamma. SymPy is easy to install and to inspect because it is written entirely in Python and it does not depend on any additional libraries. This ease of access combined with a simple and extensible code base in a well known language make SymPy a computer algebra system with a relatively low barrier of entry. SymPy includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. SymPy is free software and is licensed under "New BSD license" [25].

4.1.5 Django framework

Django [26] is a free and open source web application framework, written in Python, which follows the Model–view–controller[1] architectural pattern. It is maintained by the Django Software Foundation, an independent non-profit organization.
Django's primary goal is to ease the creation of complex, database-driven

[1]Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

websites. Django emphasizes the reusability and "pluggability" of components, rapid development, and the design pattern of non-repetitiveness. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update, and delete interfaces that are generated dynamically through introspection and configured via Django admin models [27].

4.1.6 A MATLAB differentiation matrix suite

A MATLAB differentiation matrix suite is a software suite consisting of 17 MATLAB functions for solving differential equations by the spectral collocation (i.e., pseudospectral) method. For this project three of them were needed: *chebint*, *cheb2bc* and *chebdif*.

Spectral collocation method for solving differential equations is based on interpolants of the form

$$f(x) \approx p_{N-1}(x) = \sum_{j=1}^{N} \phi_j(x) f_j. \tag{4.4}$$

$\{x_j\}_{j=1}^{N}$ is a set of distinct interpolation nodes; $f_j = f(x_j)$; and the set of cardinal functions $\{\phi_j(x)\}_{j=1}^{N}$ satisfies $\phi_j(x_k) = \delta_{jk}$ (the Kronecker delta). This means that $p_{N-1}(x)$ defined above is an interpolant of the function $f(x)$ in the sense that

$$f(x_k) = p_{N-1}(x_k), \quad k = 1, ..., N. \tag{4.5}$$

The collocation derivative operator is a concept associated with an interpolant such as (4.4). It is generated by taking ℓ derivatives of (4.4) and evaluating the result at nodes x_k:

$$f^{(\ell)}(x_k) \approx \sum_{j=0}^{N-1} \frac{d^\ell}{dx^\ell} \left[\phi_j(x)\right]_{x=x_k} f_j, \quad k = 1, ..., N. \tag{4.6}$$

The derivative operator may be represented by a differentiation matrix $D^{(\ell)}$, with entries

$$D_{k,j}^{(\ell)} = \frac{d^\ell}{dx^\ell} \left[\phi_j(x)\right]_{x=x_k}. \tag{4.7}$$

The numerical differentiation process may therefore be performed as a matrix-vector product

$$\mathbf{f}^{(\ell)} = \mathbf{D}^{(\ell)}\mathbf{f}, \tag{4.8}$$

where $\mathbf{f}$ is the the vector of function values at the nodes $\{x_k\}$.

When solving differential equations, the derivatives are approximated by the discrete derivative operators (4.8). A linear two-point boundary value problem may thus be converted into a linear system [21].

Chebdif takes two input arguments, integers N and M. N represents the size of the required differetiation matrices and M is the highest derivative needed. Output is vector x, of length N, which contains the Chebyshev points and D is an $N \times N \times M$ array containing differetiation matrices $D^{(\ell)}, \quad \ell = 1, ..., M$. It is assumed that $0 < M \leq N - 1$. Chebyshev nodes are calculated from the equation

$$x_k = \cos\left(\frac{(k-1)\pi}{N-1}\right), \quad k = 1, ..., N. \tag{4.9}$$

These are the extreme points of $T_{N-1}(x)$ [12] on $[-1, 1]$, the Chebyshev polynomial of degree N. For computing differentiation matrices *chebdif* uses interpolant given in (4.4) to calculate the set of $N - 1$ –distinct but otherwise arbitrary nodes where

$$\phi_j(x) = \frac{(-1)^j}{c_j}\frac{1-x^2}{(N-1)^2}\frac{T'_{N-1}(x)}{x - x_j}. \tag{3.18}$$

Here $c_0 = c_{N-1} = 2$ and $c_1 = ... = c_{N-2} = 1$. Differetiation matrices $D^{(\ell)}, \quad \ell = 1, ..., M$ defined in 4.7 are computed according to the following equation:

$$D^{(1)}_{k,j} = \begin{cases} \frac{c_k(-1)^{(j+k)}}{c_j(x_k - x_j)} & j \neq k \\ -\frac{1}{2}\frac{x_k}{(1-x_k^2)} & j = k \neq 1, N \\ \frac{2(N-1)^2+1}{6} & j = k = 1 \\ -\frac{2(N-1)^2+1}{6} & j = k = N. \end{cases} \qquad D^{(\ell)} = (D^{(1)})^\ell, \ell = 1, 2, ... \tag{4.10}$$

Chebint takes two arguments, the vector f of length N, contains the values of the function $f(x)$ at the Chebyshev points (4.9). The vector x, of

length 100, contains the ordinates where the interpolant is to be evaluated. On output the vector p contains the corresponding values of the interpolant $p_{N-1}(x)$ as computed by the formula[21]:

$$p_{N-1}(x) = \frac{\sum_{j=1}^{N} \frac{(-1)^j f_j}{c_j(x-x_j)}}{\sum_{j=1}^{N} \frac{(-1)^j}{c_j(x-x_j)}}. \tag{4.11}$$

Cheb2bc enables one to solve the general two-point boundary value problem of type

$$u''(x) + q(x)u'(x) + r(x)u(x) = f(x), \qquad -1 < x < 1, \tag{4.12}$$

subject to the boundary conditions

$$a_+u(1) + b_+u'(1) = c_+, \quad a_-u(-1) + b_-u'(-1) = c_-, \tag{4.13}$$

assuming that a_+ and b_+ are not both 0, and likewise for a_- and b_-. Function *cheb2bc* generates a set of nodes $\{x_k\}$, which are essentially the Chebyshev points with perhaps one or both boundary points omitted (depending on type of boundary condition). The function also returns differentiation matrices $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$ may be be computed from the Chebyshev differentiation matrices $D^{(1)}$ and $D^{(2)}$, which are computed by the *chebdif* function. The following steps are taken to solve (4.12) and (4.13):

(a) approximate $u(x)$ by the linear combination of cardinal functions $p(x)$ that satisfy the boundary conditions (4.13);

(b) require $p(x)$ to satisfy the equation (4.12) at the interpolation points, thereby converting the differential equation to a linear system;

(c) solve the linear system for the unknown function values.

The form of the interpolant in step (a) depends on the type of boundary conditions. There are three cases to consider: *Dirichlet/Dirichlet*, *Dirichlet/Robin*, and *Robin/Robin*. The interpolant is calculated based on the type of boundary condition. Details are explained below.

Dirichlet/Dirichlet Conditions.

Occurs when

$$b_+ = b_- = 0. \tag{4.14}$$

Since function values are specified at both endpoints the nodes are the interior Chebyshev points:

$$x_k = \cos\left(\frac{k\pi}{N-1}\right), \quad k = 1, ..., N-2. \tag{4.15}$$

The interpolant is a polynomial of the degree $N-1$ that satisfies the interpolation conditions

$$p_{N-1}(x_k) = u_k, \quad k = 1, ..., N-2, \tag{4.16}$$

as well as boundary conditions

$$a_+ p_{N-1}(1) = c_+, \quad a_- p_{N-1}(-1) = c_-. \tag{4.17}$$

It is given explicitly by

$$p_{N-1}(x) = \tilde{\phi}_+(x) + \tilde{\phi}_-(x) + \sum_{j=1}^{N-2} u_j \tilde{\phi}_j(x), \tag{4.18}$$

where

$$\tilde{\phi}_+(x) = \left(\frac{c_+}{a_+}\right)\phi_1(x), \tag{4.19}$$

$$\tilde{\phi}_-(x) = \left(\frac{c_-}{a_-}\right)\phi_N(x), \tag{4.20}$$

$$\tilde{\phi}_j(x) = \phi_{j+1}(x), \tag{4.21}$$

$$j = 1, ..., N-2. \tag{4.22}$$

Differetiation matrices are defined as

$$\tilde{D}^{(1)}_{k,j} = \tilde{\phi}'_j(x_k), \quad \tilde{D}^{(2)}_{k,j} = \tilde{\phi}''_j(x_k)k, \qquad j = 1, ..., N-2. \tag{4.23}$$

Since $\tilde{\phi}_j(x) = \phi_{j+1}(x)$, the matrices $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$ are the submatrices of $D^{(1)}$ and $D^{(2)}$. The function cheb2bc computes $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$ by calling chebdif

to compute $D^{(1)}$ and $D^{(2)}$ and then extracting the submatrices corresponding to rows and columns $2, ..., N-1$.

Dirichlet/Robin Conditions.

Let $b_+ \neq 0$ and $b_- = 0$. In this case there is a Dirichlet condition at $x = -1$ and a Robin condition at $x = 1$. Since function values is specified at $x = -1$ the Chebyshev node corresponding to $k = N$ is dropped:

$$x_k = \cos\left(\frac{(k-1)\pi}{N-1}\right), \quad k = 1, ..., N-1. \tag{4.24}$$

The interpolant is a polynomial of a degree N that satisfies the interpolation conditions

$$p_N(x_k) = u_k, \quad k = 1, ..., N-1, \tag{4.25}$$

as well as boundary conditions

$$a_+ p_N(1) + b_+ p_N^{'}(1) = c_+, \quad a_- p_N(-1) = c_-. \tag{4.26}$$

It is given explicitly by

$$p_N(x) = \tilde{\phi}_+(x) + \tilde{\phi}_-(x) + \sum_{j=1}^{N-1} u_j \tilde{\phi}_j(x), \tag{4.27}$$

where

$$\tilde{\phi}_+(x) = \left(\frac{c_+}{b_+}\right)(x-1)\phi_1(x), \tag{4.28}$$

$$\tilde{\phi}_-(x) = \left(\frac{c_-}{a_-}\right)\left(\frac{1-x}{2}\right)\phi_N(x), \tag{4.29}$$

$$\tilde{\phi}_1(x) = \left(1 - \left(\phi_1^{'} + \frac{a_+}{b_+}\right)(x-1)\right)\phi_1(x), \tag{4.30}$$

$$\tilde{\phi}_j(x) = \left(\frac{1-x}{1-x_j}\right)\phi_{j+1}(x), \quad j = 2, ..., N-1. \tag{4.31}$$

Differetiation matrices are defined as

$$\tilde{D}^{(1)}_{k,j} = \tilde{\phi}_j^{'}(x_k), \quad \tilde{D}^{(2)}_{k,j} = \tilde{\phi}_j^{''}(x_k), \qquad k, j = 1, ..., N-1. \tag{4.32}$$

The quantities $\tilde{\phi}_j^{'}(x_k)$ and $\tilde{\phi}_j^{''}(x_k)$ may be expressed explicitly in terms of quantities $\phi_j^{'}(x_k)$ and $\phi_j^{''}(x_k)$, which are the entries of the standard Chebyshev differentiation matrices $D^{(1)}$ and $D^{(2)}$ which are computed by chebdif function.

Robin/Robin Conditions.

Occurs when $b_+ \neq 0$ and $b_- \neq 0$. Since function values are specified at neither endpoint the nodes are the full set of Chebyshev points:

$$x_k = \cos\left(\frac{(k-1)\pi}{N-1}\right), \quad k = 1, ..., N. \tag{4.33}$$

The interpolant is a polynomial of degree $N+1$ that satisfies the interpolation conditions

$$p_{N+1}(x_k) = u_k, \quad k = 1, ..., N, \tag{4.34}$$

as well as the boundary conditions

$$a_+ p_{N+1}(1) + b_+ p_{N+1}^{'}(1) = c_+, \quad a_- p_{N+1}(-1) + b_- p_{N+1}^{'}(-1) = c_-. \tag{4.35}$$

It is given explicitly by

$$p_{N+1}(x) = \tilde{\phi}_+(x) + \tilde{\phi}_-(x) + \sum_{j=1}^{N} u_j \tilde{\phi}_j(x), \tag{4.36}$$

where

$$\tilde{\phi}_+(x) = \left(\frac{c_+}{b_+}\right)\left(\frac{(x^2-1)}{2}\right)\phi_1(x), \tag{4.37}$$

$$\tilde{\phi}_-(x) = \left(\frac{c_-}{b_-}\right)\left(\frac{1-x^2}{2}\right)\phi_N(x), \tag{4.38}$$

$$\tilde{\phi}_1(x) = \left[\frac{1+x}{2} + \left(\frac{1}{2} + \phi_1^{'}(1) + \frac{a_+}{b_+}\right)\left(\frac{1-x^2}{2}\right)\right]\phi_1(x), \tag{4.39}$$

$$\tilde{\phi}_j(x) = \left(\frac{1-x^2}{1-x_j^2}\right)\phi_j(x), \quad j = 2, ..., N-1, \tag{4.40}$$

$$\tilde{\phi}_N(x) = \left[\frac{1-x}{2} + \left(\frac{1}{2} - \phi_N^{'}(-1) + \frac{a_-}{b_-}\right)\left(\frac{1-x^2}{2}\right)\right]\phi_N(x). \tag{4.41}$$

Differetiation matrices are defined as

$$\tilde{D}_{k,j}^{(1)} = \tilde{\phi}_j^{'}(x_k), \quad \tilde{D}_{k,j}^{(2)} = \tilde{\phi}_j^{''}(x_k), \qquad k, j = 1, ..., N. \tag{4.42}$$

[21]

The function *cheb2bc* computes the various matrices and boundary condition vectors described above. It takes two arguments N and g where g is an array of coefficients $a_+, b_+, c_+, a_-, b_-, c_-$ and outputs the node vector x, matrices $D1t$, $D2t$, *phip* and *phim*. $D1t$ and $D2t$ contain $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$, respectively. The first and second columns of *phip* contain $\tilde{\phi}'_+(x)$ and $\tilde{\phi}''_+(x)$, evaluated at the points in the node vector. Similarly, the first and second columns of *phim* contain $\tilde{\phi}'_-(x)$ and $\tilde{\phi}''_-(x)$, evaluated at the points in the node vector. Since $\tilde{\phi}_+(x)$ and $\tilde{\phi}_-(x)$ are both 0 at points in the node vector, their values are not returned. The values of output variables (beside x) vary depending on a types of boundary conditions defined earlier in this subsection.

4.2 Computer program for solving two-point boundary value problems

The program for solving two-point boundary value problems is written as a Django application and incorporated in the Django project. The input consists of N, $q(x)$, $r(x)$, $f(x)$ and an array of coefficients g. The array of coefficients g is put together from boundary condition's a_+, b_+, $, c_+$ and a_-, b_-, c_- coefficients. After pressing the button "Calculate" on the first page of the project, the program outputs solution graph and user inputs (in form of a mathematical annotation generated by Latex online tool[2]) and equation on a page labeled "/result". The whole result page, together with solution graph, is generated on the server and sent to the client in form of HTTP response object with the solution graph coded to "base64" format. The general form of equation that the user inputs is defined in the beginning of this chapter and input parameters are explained in the subsection "A MATLAB differentiation matrix suite". Examples of how the program works are given in the subsection below.

[2]http://www.codecogs.com/latex/eqneditor.php

4.2.1 Examples

First example is equation of a form $u''(x) + xu'(x) = (2 + x^2)\cos(x)$, with boundary conditions $u(-1) = sin(1), u(1) = sin(1)$ with solution $u(x) = xsin(x)$.

Second example is equation of a form $u''(x) - |x|u'(x) + 2u(x) = 30|x|x^3 + 2|x|x^5 - 6x^6$, with boundary conditions $u(-1) = -1, u(1) = 1$ with solution $u(x) = |x|x^5$.

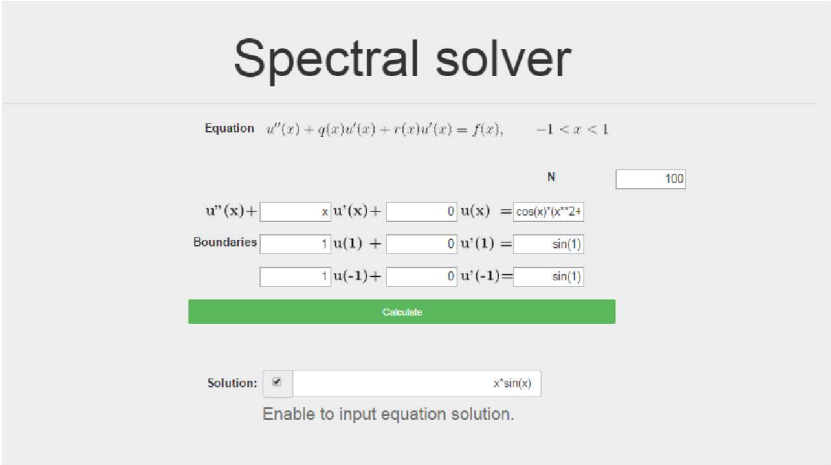

Figure 4.1: Example of program input 1

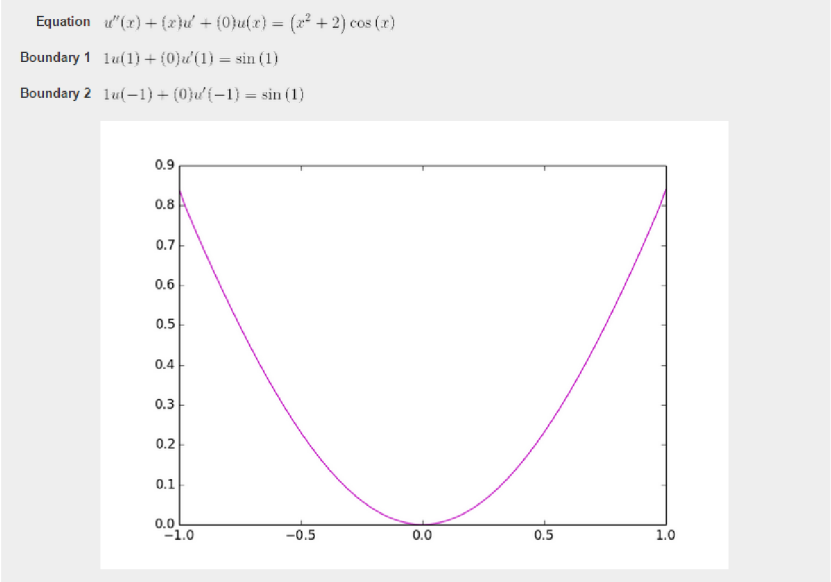

Figure 4.2: Example of program output 1

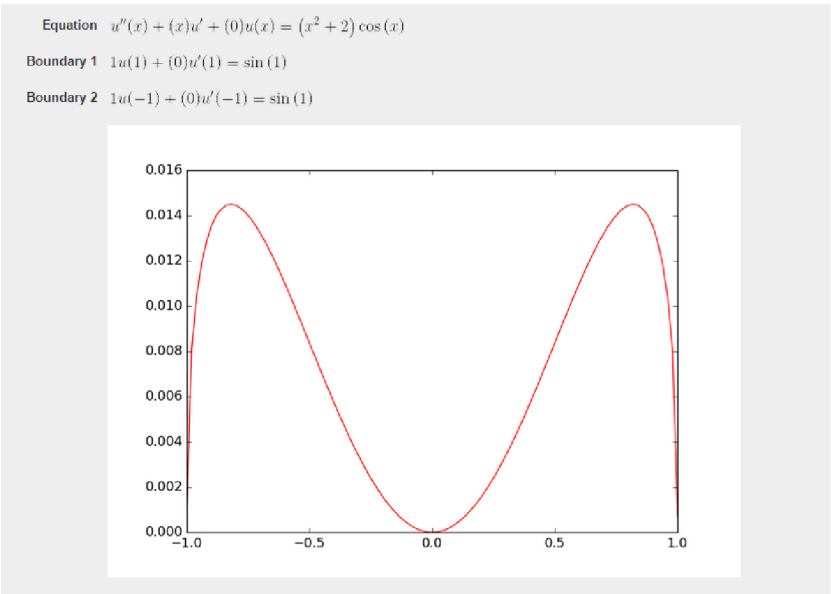

Figure 4.3: Error in first calculation

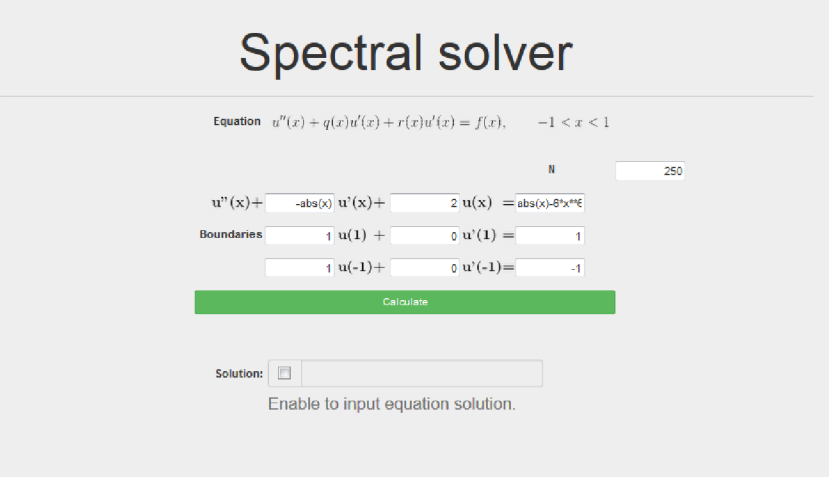

Figure 4.4: Example of program input 2

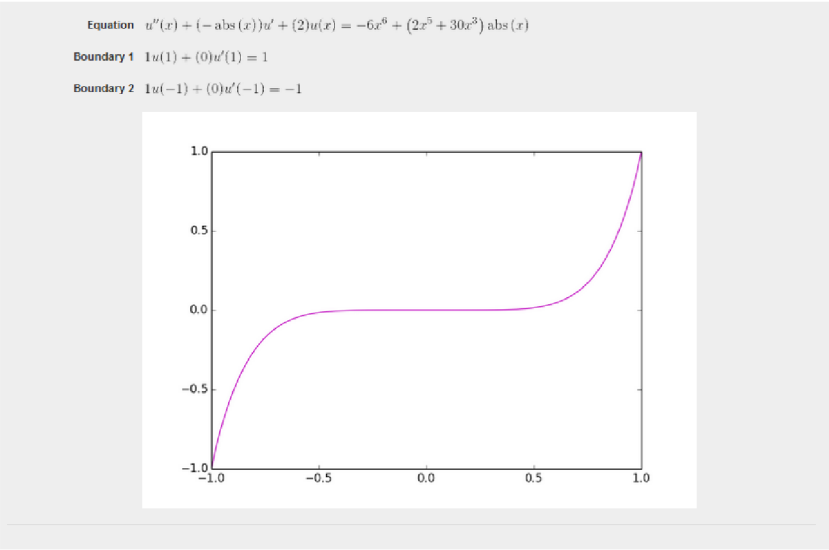

Figure 4.5: Example of program output 2

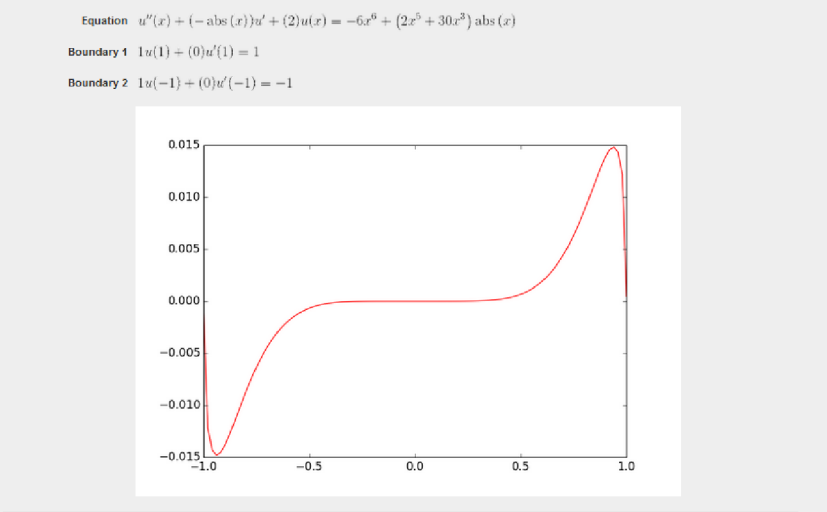

Figure 4.6: Error in second calculation

Chapter 5

Conclusion

The aim of this thesis was to study a problem from the field of numerical mathematics and solve it using programming language of choice while using skills learned in the past few years of studies. Together with my mentor we have decided to tackle pseudospectral methods for two point value problems. The result was a user friendly web application used to approximate solution of two point boundary value problems. The application is written with Python programming language using Django, Scipy, Numpy and Sympy. The most challenging part was to write a piece of code that works for all equations.

During the course of this research one algorithm and several concepts were learned and our memories were refreshed in connection to differential equations as well as boundary value problems.

Bibliography

[1] Carrier, George F., and Pearson, Carl E.. "Partial differential equations: theory and technique." Academic Press, 2014.

[2] Weisstein, Eric W., "Partial Differential Equation." From MathWorld –A Wolfram Web Resource.
Retrieved July 22, 2015, from: http://mathworld.wolfram.com/PartialDifferentialEquation.html

[3] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/Differential_equation

[4] Weisstein, Eric W. "Differential Operator." From MathWorld –A Wolfram Web Resource.
http://mathworld.wolfram.com/DifferentialOperator.html

[5] Hadamard, J. "Sur les problmes aux drives partielles et leur signification physique." Princeton university bulletin 13.49-52 (1902): 28.

[6] Duffy D. G. "Mixed boundary value problems." CRC Press, 2008.

[7] Swati C. ,Hosch W. L. "boundary value" Retrieved July 22, 2015, from: http://www.britannica.com/EBchecked/topic/75656/boundary-value

[8] Epperson, James F. "An introduction to numerical methods and analysis. John Wiley & Sons, 2013.

[9] Mason, John C., and David C. Handscomb. "Chebyshev polynomials". CRC Press, 2002.

[10] Schlatter P. “Spectral Methods” Retrieved July 23, 2014, from: http://www.mech.kht.se/ãrdeshir/courses/literature/Notes_Spectral_Methods.pdf

[11] Boyd, John P. “Chebyshev and Fourier spectral methods”. Courier Dover Publications, 2001.

[12] Abramowitz, Milton, and Irene A. Stegun. ”Handbook of mathematical functions with formulas, graphs, and mathematical tables.” Conference on XYZ. Dover, 2006.

[13] Runge, Carl. “ber empirische Funktionen und die Interpolation zwischen quidistanten Ordinaten.” Zeitschrift fr Mathematik und Physik 46.224-243 (1901): 20.

[14] Davis, Philip J. Interpolation and approximation. Courier Dover Publications, 1975.

[15] Gottlieb, David, and Steven A. Orszag. “Numerical Analysis of Spectral Methods: Theory and Applications”. Vol. 26. SIAM, 1977.

[16] Fox, Leslie, and Ian Bax Parker. “Chebyshev polynomials in numerical analysis.” , 1968.

[17] Retrieved July 22, 2015, from: https://www.python.org/

[18] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/Python_programming_ language

[19] Retrieved July 22, 2015, from: http://scipy.org/

[20] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/SciPy

[21] Weideman, J. Andre, and Satish C. Reddy. “A MATLAB differentiation matrix suite.” ACM Transactions on Mathematical Software (TOMS) 26.4 (2000): 465-519.

[22] Retrieved July 22, 2015, from: http://www.numpy.org/

[23] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/NumPy

[24] Retrieved July 22, 2015, from: http://www.sympy.org/en/index.html

[25] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/SymPy

[26] Retrieved July 22, 2015, from: https://www.djangoproject.com/

[27] Retrieved July 22, 2015, from: http://en.wikipedia.org/wiki/Django_ web_ framework